Incremental Data Flow analysis using PRISM

Rashmi Rekha Mech
(*Project Guide: Prof. Uday Khedker*)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

Oct 2014

Outline of the talk

- Incremental Data Flow analysis
  - Bit-vector framework
  - Constant Propagation analysis
- Overview of PRISM
- Liveness-based Reaching Definition analysis
  - Performance measurement
- Future work

Part I

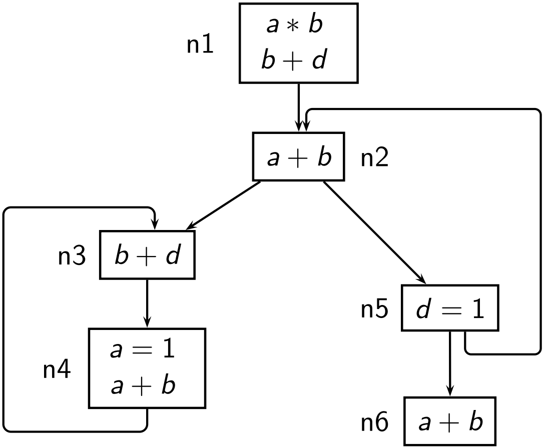
Incremental Data Flow analysis

Why Incremental Analysis?

When program undergoes changes:

- Some or all computed data flow information become invalid
- Recompution is required

Motivating Example - Available Expression Analysis

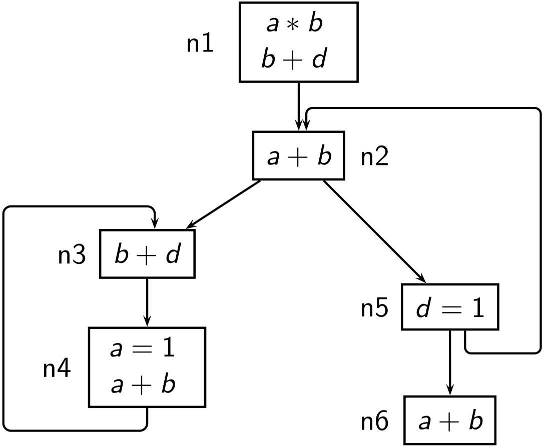


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration



n1 $a * b$ $b + d$

n2 $a + b$

n3 $b + d$

n4 $a = 1$ $a + b$

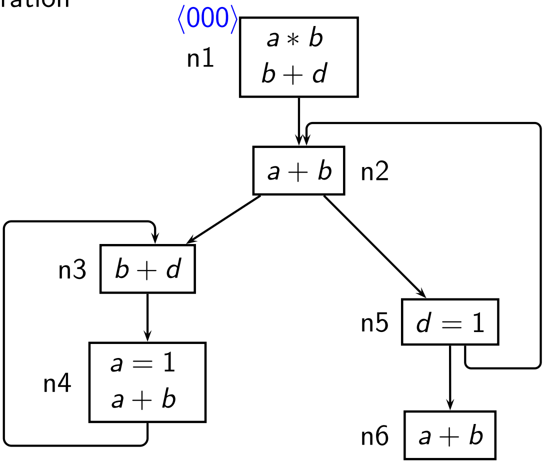
n5 $d = 1$

n6 $a + b$

Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration



$\langle 000 \rangle$

n1 $a * b$ $b + d$

n2 $a + b$

n3 $b + d$

n4 $a = 1$ $a + b$

n5 $d = 1$

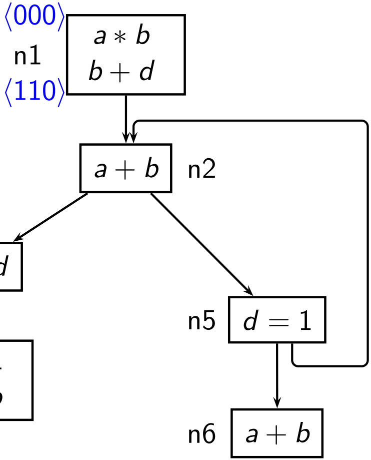
n6 $a + b$

Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration

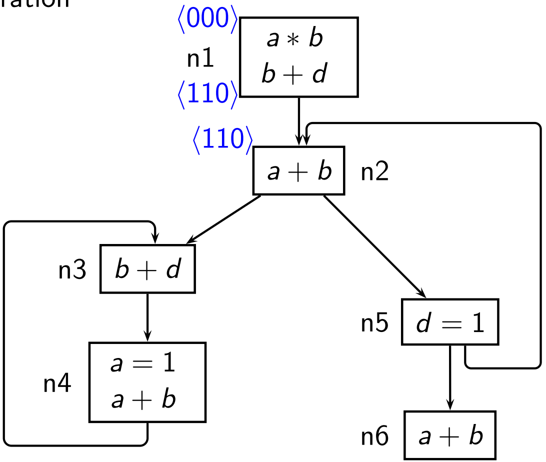


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

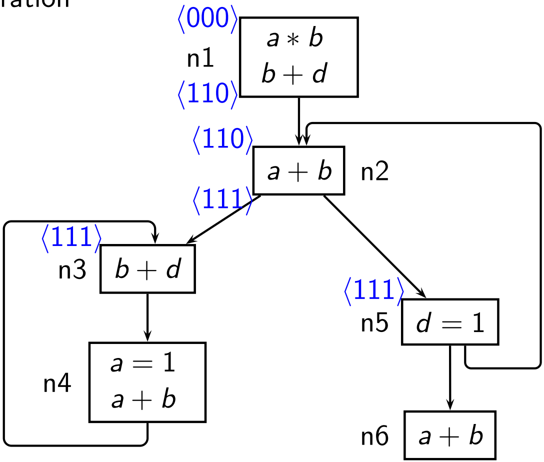
1st Iteration



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration



⟨000⟩

n1

$a * b$

$b + d$

⟨110⟩

⟨110⟩

$a + b$

n2

⟨111⟩

⟨111⟩

n3

$b + d$

n4

$a = 1$

$a + b$

⟨111⟩

n5

$d = 1$

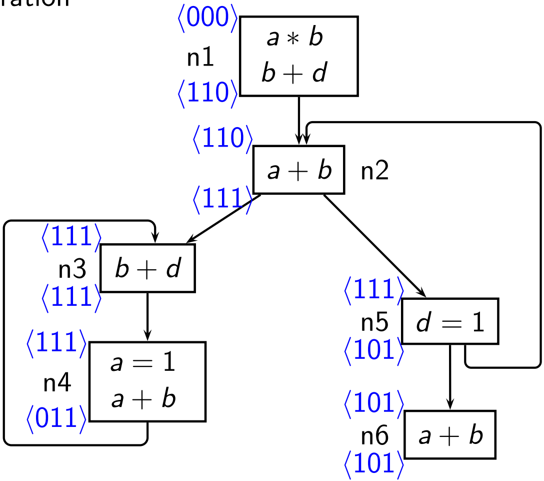
n6

$a + b$

Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration

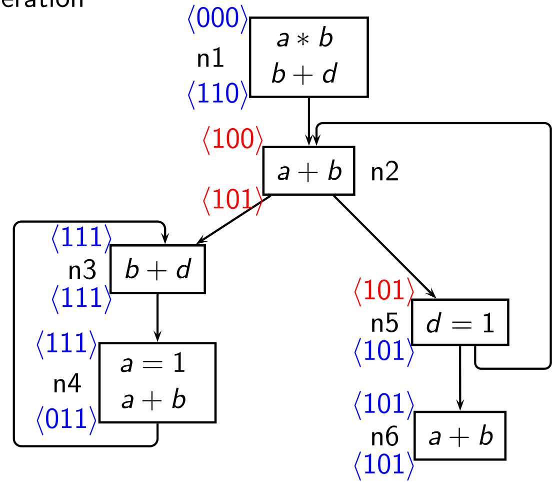


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

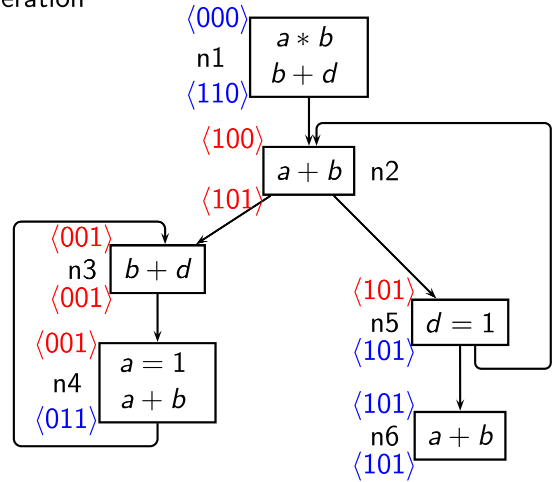
2nd Iteration



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

2nd Iteration

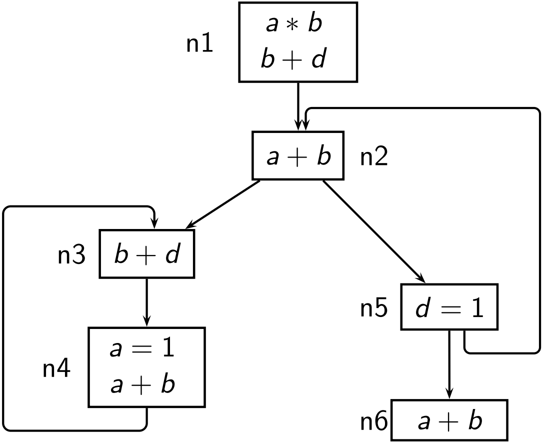


Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

- It requires 3 iterations to converge

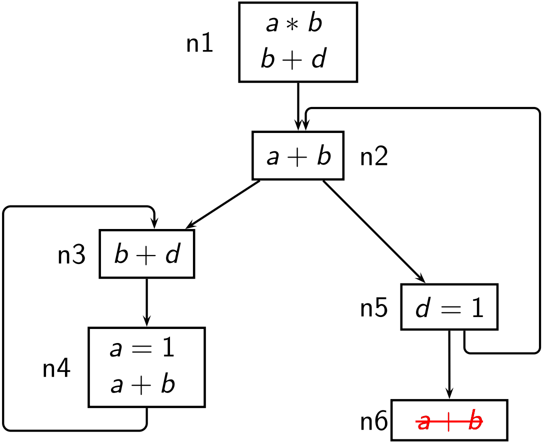
Motivating Example - Available Expression Analysis



Bit Vector

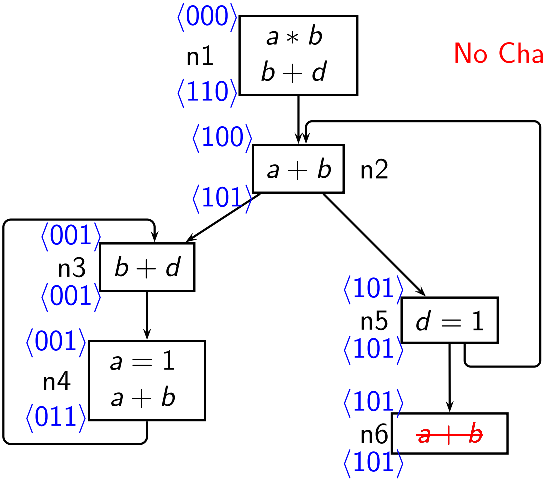
$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis



No Change in data flow values

Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

- Recomputing the values from the scratch is very inefficient
- Need an incremental analysis:
    - modifies only affected data flow information
    - more cost effective then **exhaustive** analysis

Part II

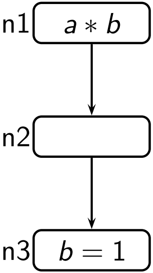
Incremental analysis in Bit-vector framework

Flow functions in Bit-vector framework

- Possible flow functions:
    - Raise : Results is always Top
    - Lower : Results is always Bottom
    - Propagate : Propagates the value from one program point to another

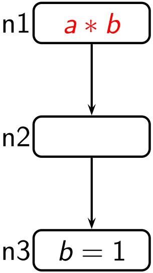
Example for flow functions

Available Expression Analysis



Example for flow functions

Available Expression Analysis

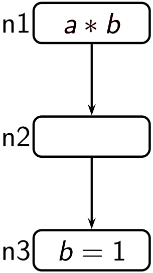


Raise Function

$\text{Gen}_1 = 1$
$\text{Kill}_1 = 0$
$\text{IN}_1 = 0$
$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1 \text{-} \text{Kill}_1) = 1$

Example for flow functions

Available Expression Analysis



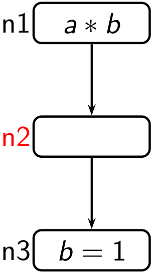
Raise Function

$\text{Gen}_1 = 1$
$\text{Kill}_1 = 0$
$\text{IN}_1 = 0$
$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1\text{-Kill}_1) = 1$

Result is always top

Example for flow functions

Available Expression Analysis



Propagate Function

$\text{Gen}_2 = 0$

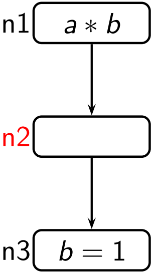
$\text{Kill}_2 = 0$

$\text{IN}_2 = 1$

$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2\text{-Kill}_2) = \text{IN}_2 = 1$

Example for flow functions

Available Expression Analysis



Propagate Function

$\text{Gen}_2 = 0$

$\text{Kill}_2 = 0$

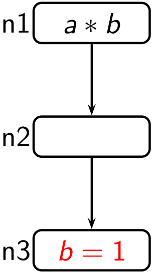
$\text{IN}_2 = 1$

$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2\text{-Kill}_2) = \text{IN}_2 = 1$

Propagates the value at IN to OUT

Example for flow functions

Available Expression Analysis

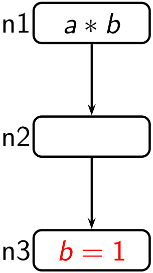


Lower Function

Gen$_3$ $= 0$
Kill$_3$ $= 1$
IN$_3$ $= 1$
OUT$_3$ $=$Gen$_3$$\cup$ (IN$_3$-Kill$_3$) $= 0$

Example for flow functions

Available Expression Analysis



Lower Function

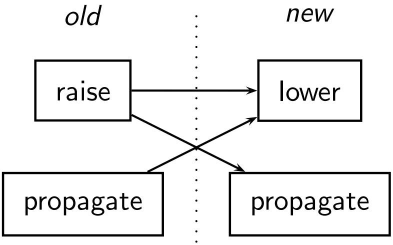
$\text{Gen}_3 = 0$
$\text{Kill}_3 = 1$
$\text{IN}_3 = 1$
$\text{OUT}_3 = \text{Gen}_3 \cup (\text{IN}_3\text{-Kill}_3) = 0$

Result is always bottom

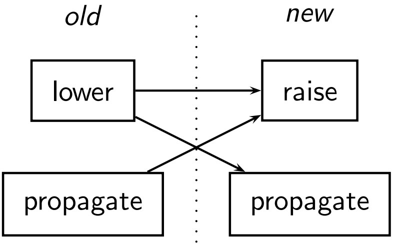
Changes in Bit-vector framework

- As a consequence of some change in a node, some data flow values may:
  - change from top to bottom
  - change from bottom to top
  - remain same

Possible changes in flow functions for top to bottom change



Possible changes in flow functions for bottom to top change

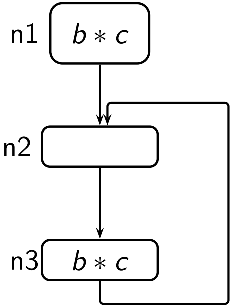


Handling Top to Bottom change

- Top value is an intermediate value until data flow analysis is completed
- Whenever there is top to bottom change, the changes can be propagated directly to its neighbouring nodes

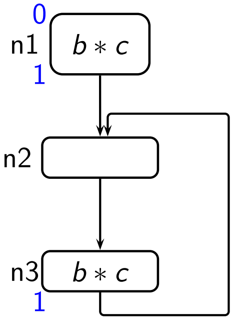
Example for Top to Bottom change

Initial Available Expression Analysis



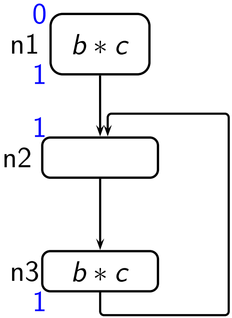
Example for Top to Bottom change

Initial Available Expression Analysis



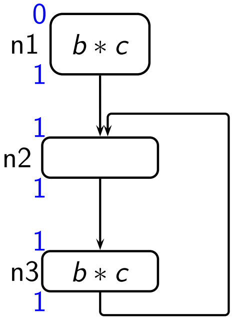
Example for Top to Bottom change

Initial Available Expression Analysis

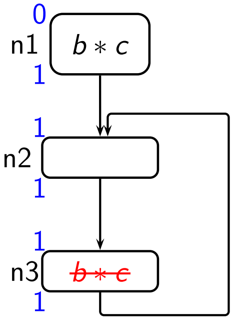


Example for Top to Bottom change

Initial Available Expression Analysis

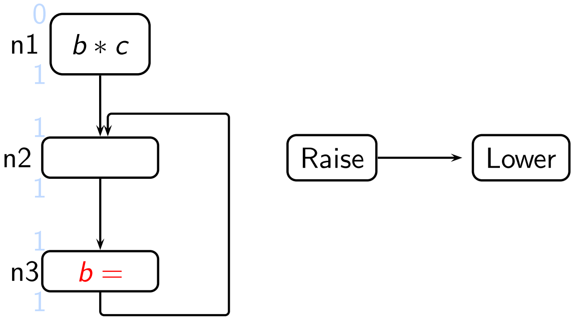


Example for Top to Bottom change



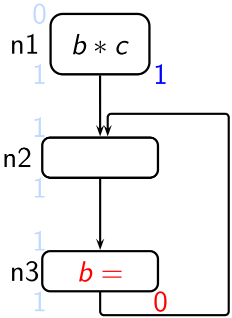
Example for Top to Bottom change

Top to Bottom change



Example for Top to Bottom change

Top to Bottom change



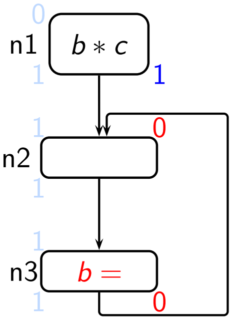
$\text{IN}_2 = \text{OUT}_1 \cap \text{OUT}_3$

Raise → Lower

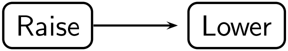
Directly Propagate the change to its neighbour

Example for Top to Bottom change

Top to Bottom change



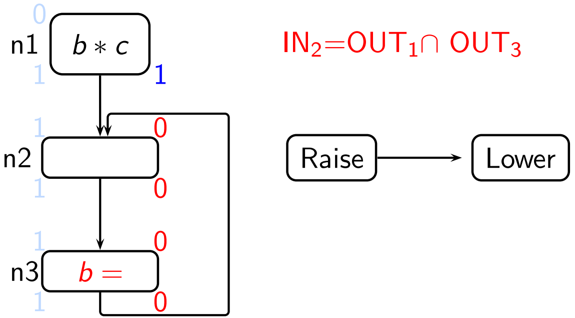
$\text{IN}_2 = \text{OUT}_1 \cap \text{OUT}_3$



Directly Propagate the change to its neighbour

Example for Top to Bottom change

Top to Bottom change



$IN_2 = OUT_1 \cap OUT_3$

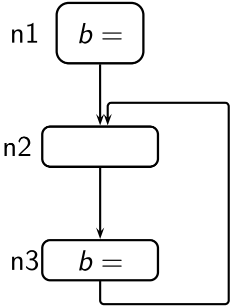
Directly Propagate the change to its neighbour

Handling Bottom to Top change

- Bottom value is a final value even during analysis
- Whenever there is bottom to top change, we cannot directly propagate the changes to its neighbouring nodes
- Need some more processing

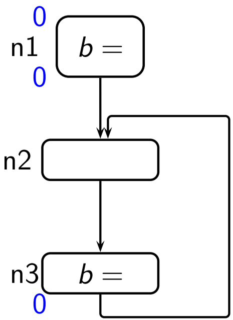
Example for Bottom to Top change

Initial Available Expression Analysis



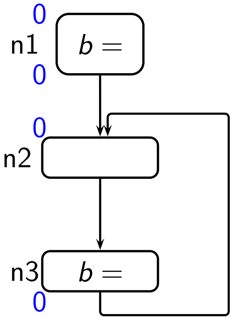
Example for Bottom to Top change

Initial Available Expression Analysis



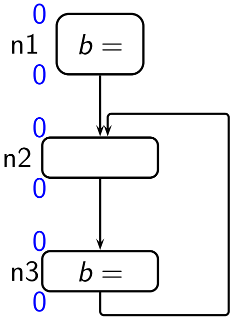
Example for Bottom to Top change

Initial Available Expression Analysis

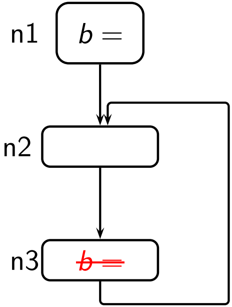


Example for Bottom to Top change

Initial Available Expression Analysis

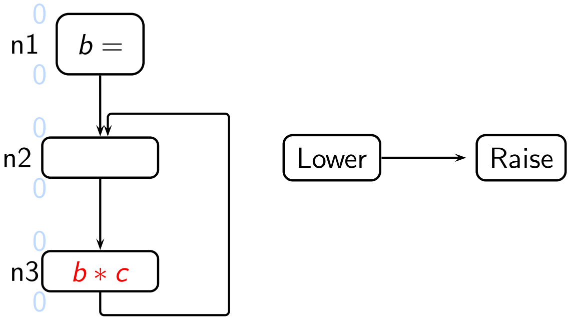


Example for Bottom to Top change



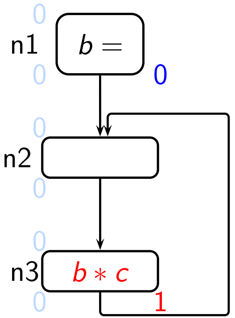
Example for Bottom to Top change

Bottom to Top change

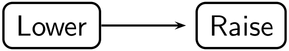


Example for Bottom to Top change

Bottom to Top change



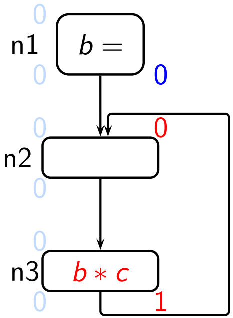
$IN_2 = OUT_1 \cap OUT_3$



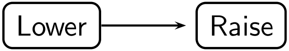
Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top change

Bottom to Top change



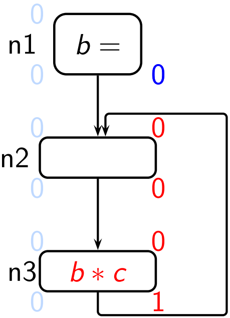
$IN_2 = OUT_1 \cap OUT_3$



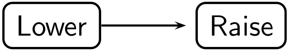
Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top change

Bottom to Top change



$IN_2 = OUT_1 \cap OUT_3$


Lower
Raise

Cannot propagate the change to its neighbouring nodes

Bottom to Top change

- Need some more processing

Bottom to Top change

- Steps to incorporate Bottom to Top change:

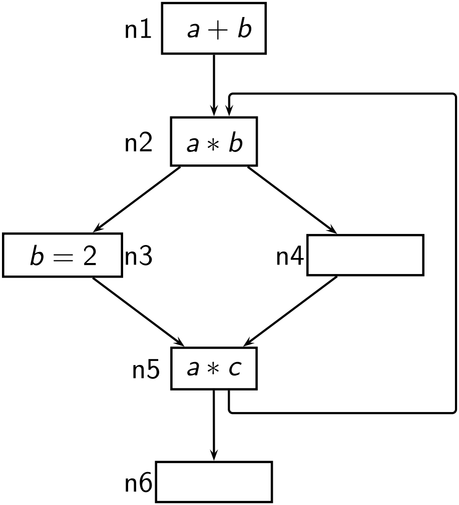
Bottom to Top change

- Steps to incorporate Bottom to Top change:
    - Identify the data flow values which may becomes top

Bottom to Top change

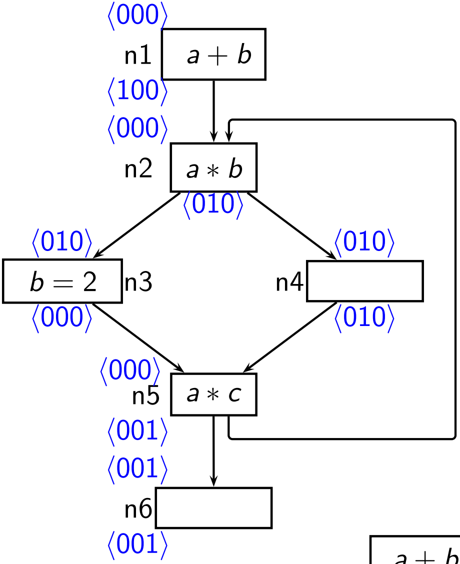
- Steps to incorporate Bottom to Top change:
  - Identify the data flow values which may becomes top
  - Find out the data flow values which must remain bottom due to the effect of some other property

Motivating Example



$a + b$	$a * b$	$a * c$

Motivating Example

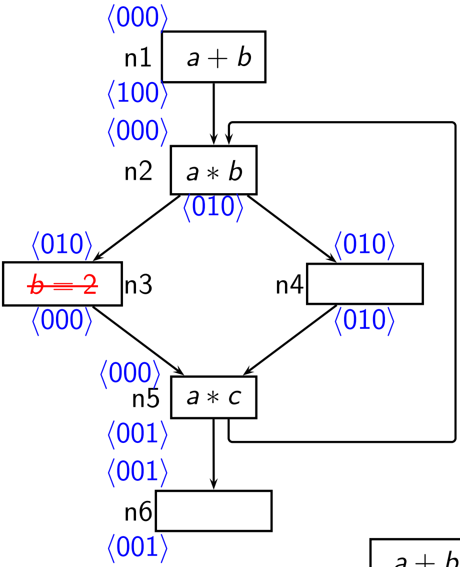


Initial Available Expression Analysis

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a+b$	$a*b$	$a*c$

Motivating Example

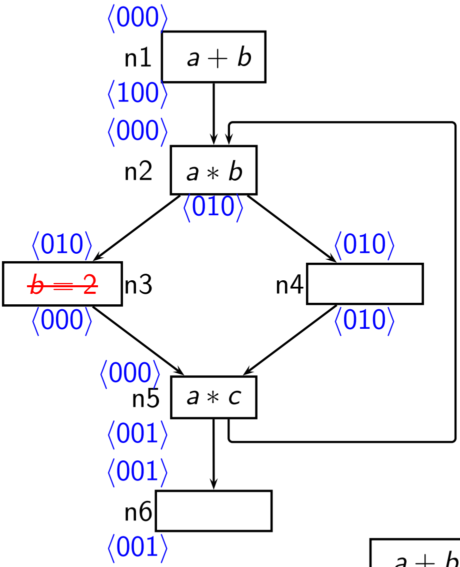


Initial Available Expression Analysis

	$a + b$		$a * b$		$a * c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a + b$	$a * b$	$a * c$

Motivating Example



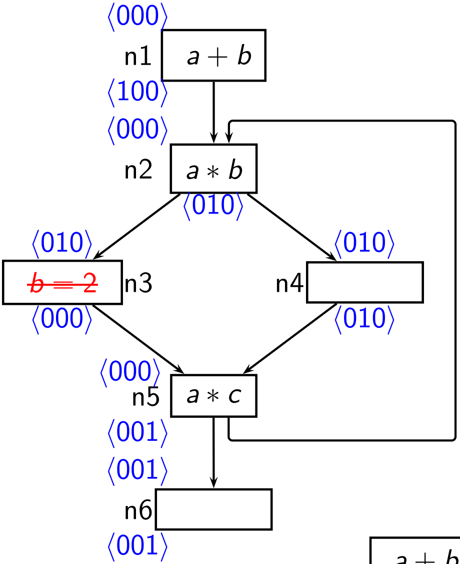
Initial Available Expression Analysis

lower

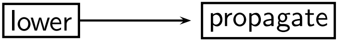
propagate

$a + b$	$a * b$	$a * c$

Motivating Example

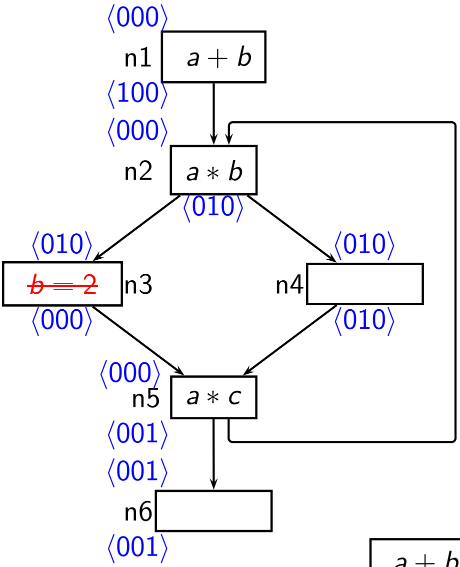


Initial Available Expression Analysis

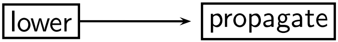


$a + b$	$a * b$	$a * c$

Motivating Example



Initial Available Expression Analysis



Bottom to Top change

$a + b$	$a * b$	$a * c$

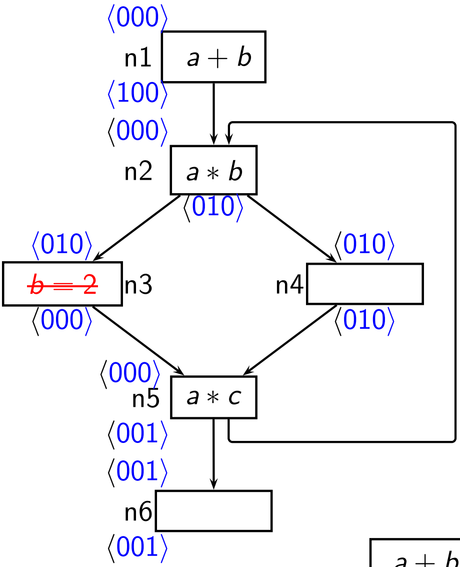
Motivating Example - Step 1

- The data flow values which were 0 and *may* become 1 due to this change

Motivating Example - Step 1

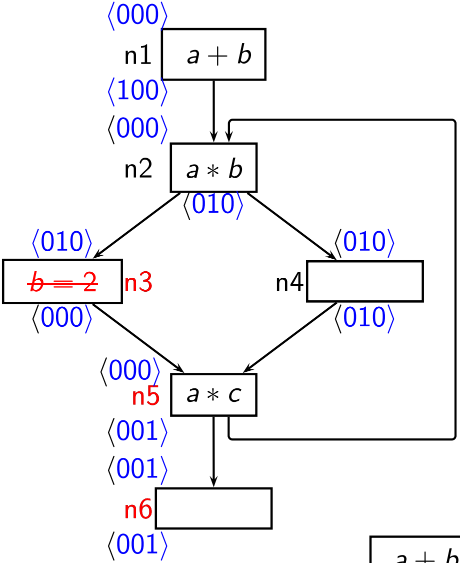
- The data flow values which were 0 and *may* become 1 due to this change
    - Affected region

Motivating Example - Step 1



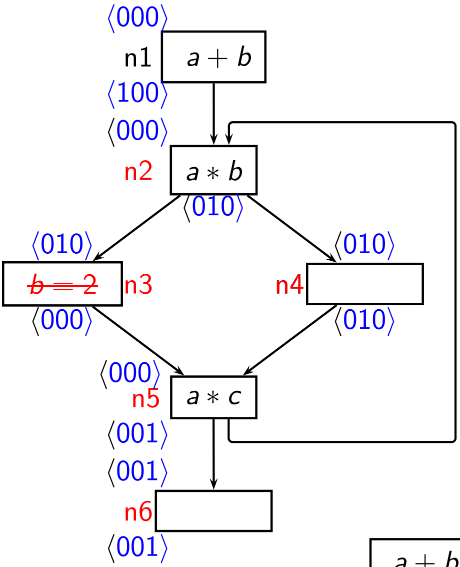
$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



$a + b$	$a * b$	$a * c$

Motivating Example - Step 1

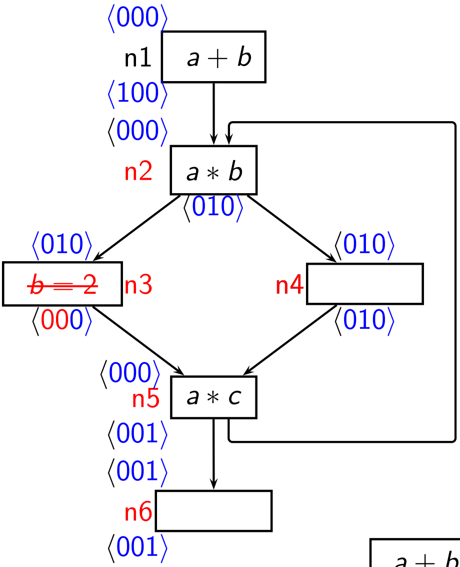


Affected Region

$\langle$ OUT$_3$ IN$_5$ OUT$_5$, IN$_6$, OUT$_6$, IN$_2$, OUT$_2$, IN$_4$, OUT$_4$, IN$_3\rangle$

$a + b$	$a * b$	$a * c$

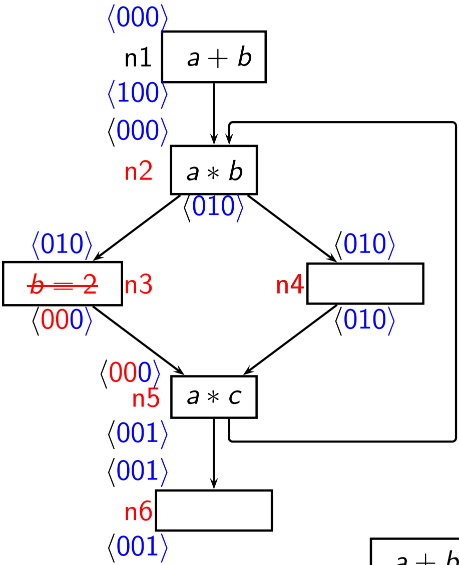
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

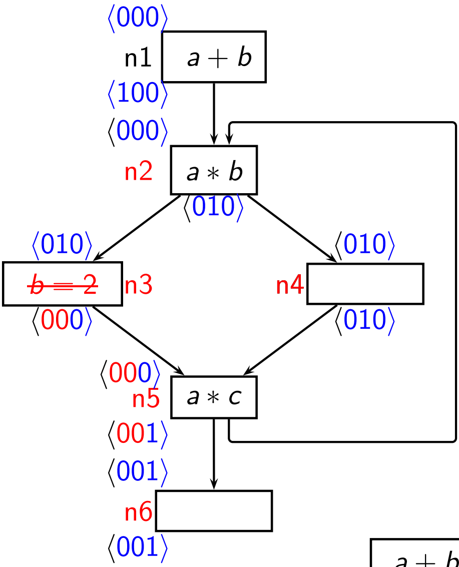
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

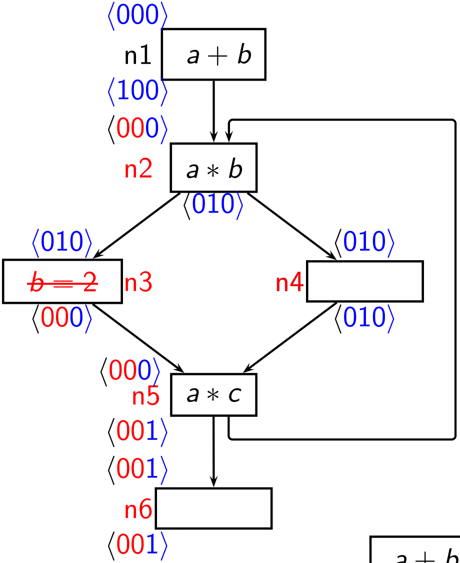
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

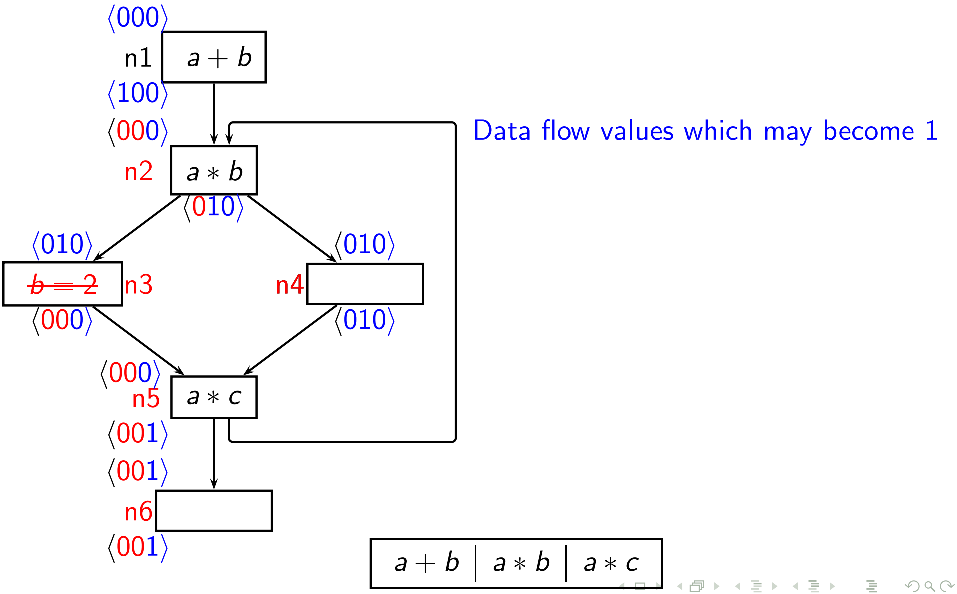
Motivating Example - Step 1



Data flow values which may become 1

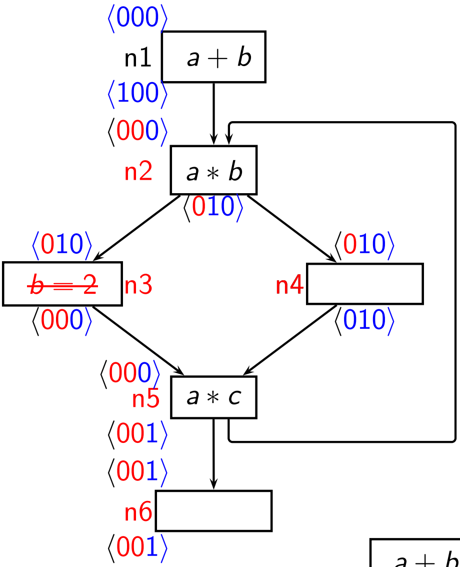
$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



Data flow values which may become 1

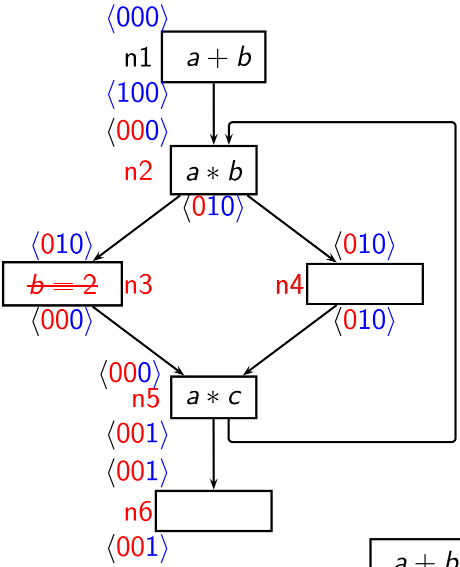
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

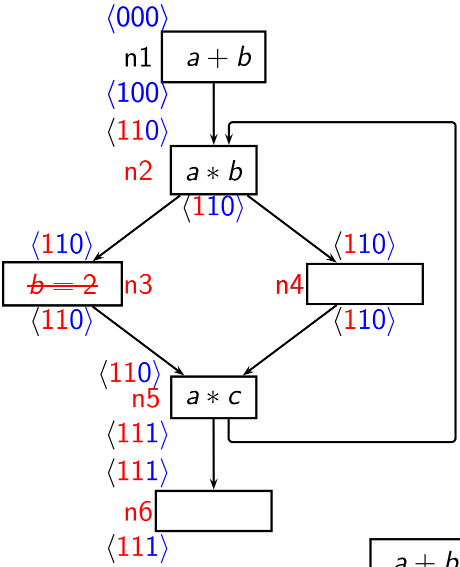
Motivating Example - Step 1



Data flow values which may become 1

a + b	a * b	a * c

Motivating Example - Step 1



Data flow values which may become 1

	$a + b$		$a * b$		$a * c$	
Node	In	Out	In	Out	In	Out
1.						
2.	1	1	1			
3.	1	1		1		
4.	1	1				
5.	1	1	1	1		
6.	1	1	1	1		

$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property

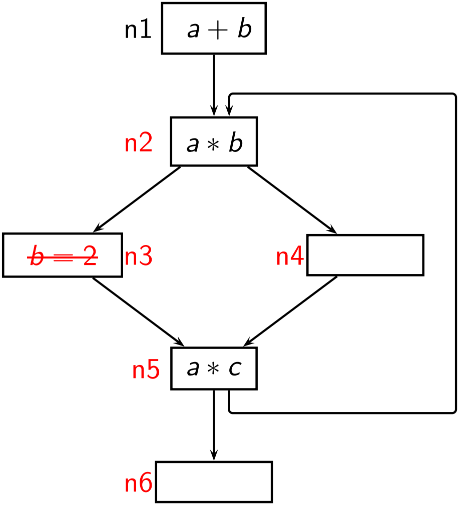
Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property
    - Identifying Boundary nodes

Motivating Example - Step 2

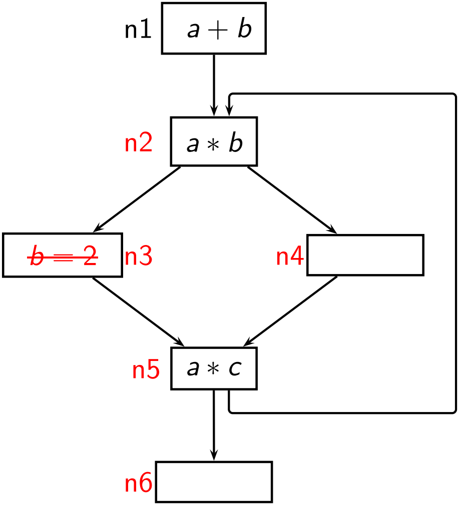
- Find out the data flow values which must remain bottom due to the effect of some other property
  - Identifying Boundary nodes
  - Computing values at boundary nodes and propagating them

Motivating Example - Step 2



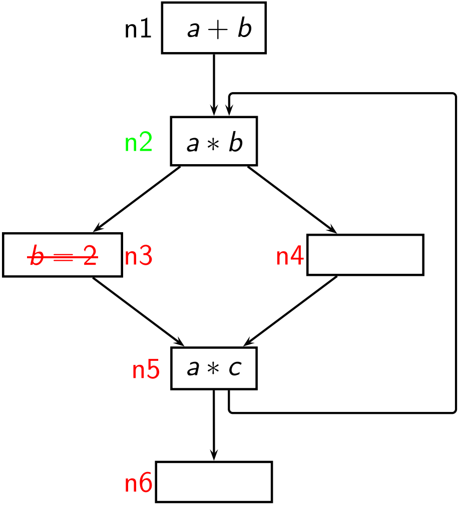
$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

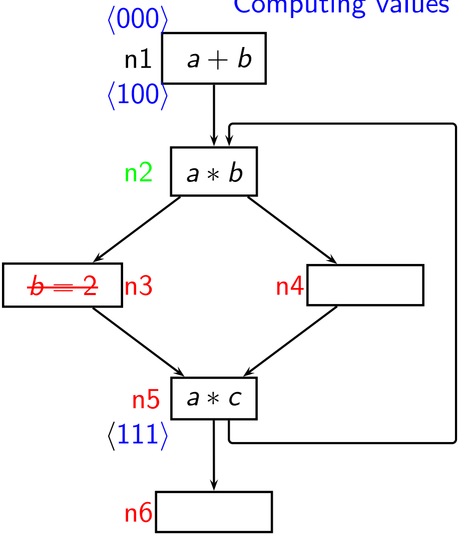


Node 2 is Boundary node

$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

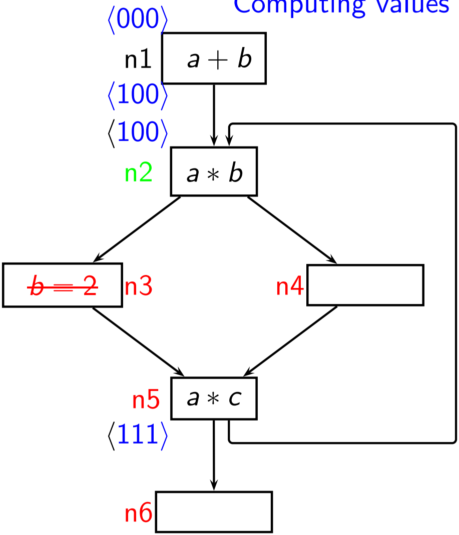
Computing values at boundary node and propagating them



$a + b$	$a * b$	$a * c$

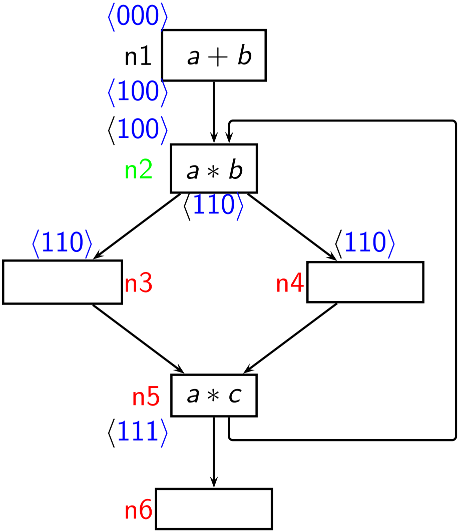
Motivating Example - Step 2

Computing values at boundary node and propagating them



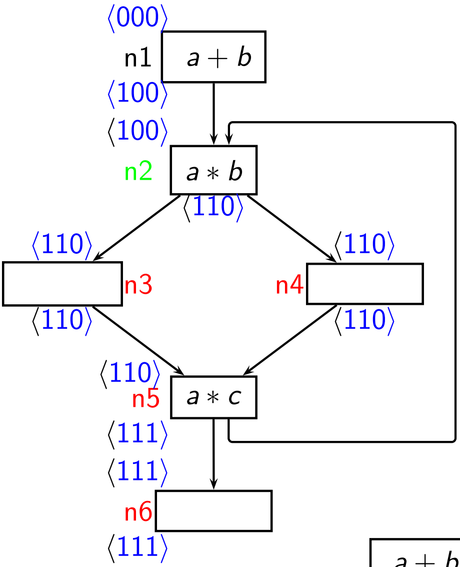
$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



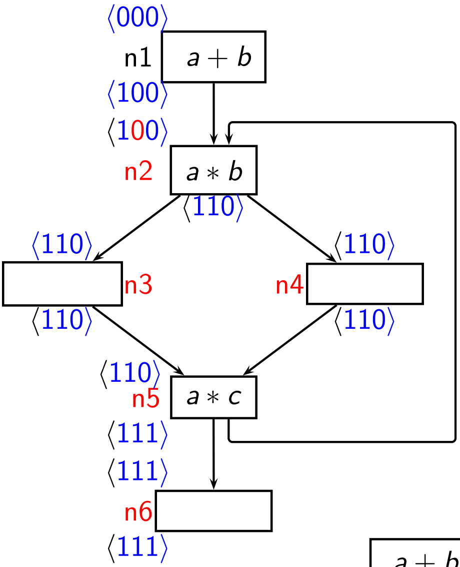
$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

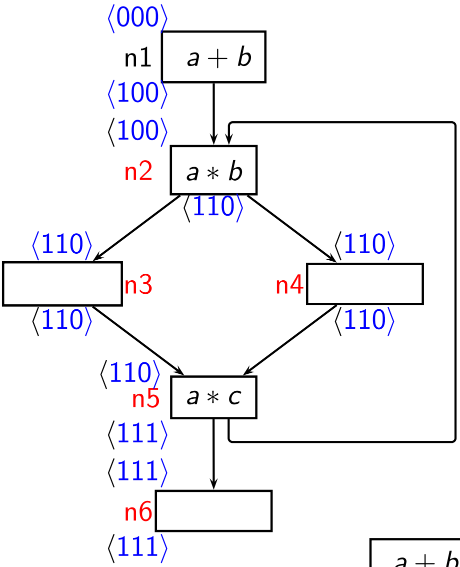


Values which must remain 0

	$a + b$		$a * b$		$a * c$	
Node	In	Out	In	Out	In	Out
1.						
2.			0			
3.						
4.						
5.						
6.						

$a + b$	$a * b$	$a * c$

Motivating Example - Step 2



Final values

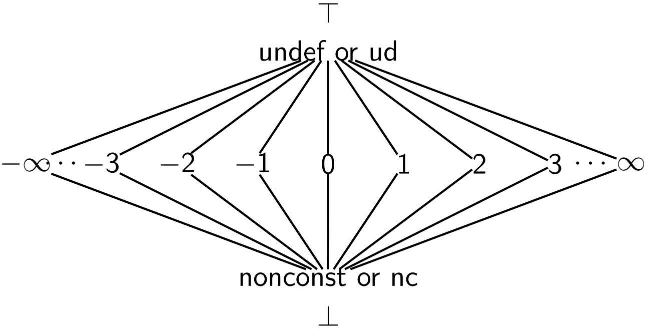
	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	1	1	0	1	0	0
3.	1	1	1	1	0	0
4.	1	1	1	1	0	0
5.	1	1	1	1	0	1
6.	1	1	1	1	1	1

$a+b$	$a*b$	$a*c$

Part III

Incremental Analysis in Constant Propagation

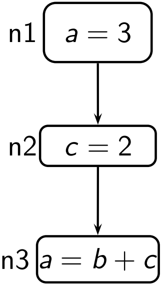
Component lattice for Constant Propagation



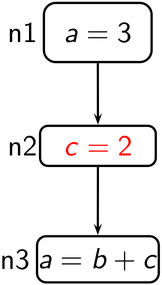
Flow functions

- Possible flow functions
    - Top : Similar to raise function
    - Bottom : Similar to lower function
    - Constant : Always produce a constant value
    - Side level : Result depends on the operands of the expression

Constant functions

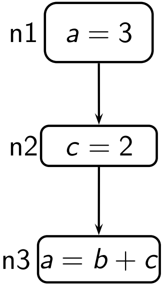


Constant functions

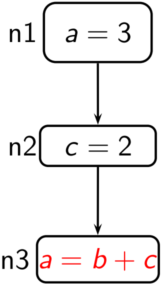


Produces Constant values

Side level functions



Side level functions

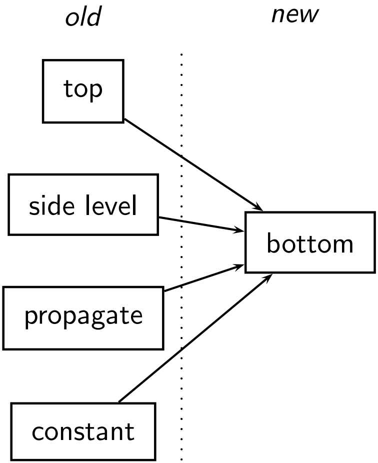


Result depends on the operands

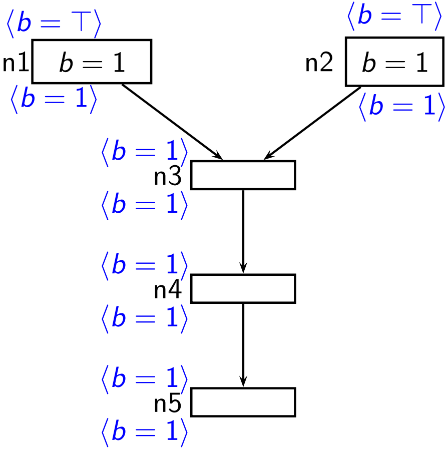
Issues in Constant propagation

- When there is a change to bottom:
    - No need of creating affected region
- Otherwise, need to create affected region

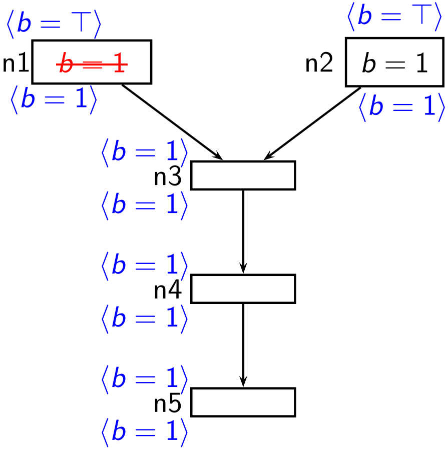
Issues in Constant propagation



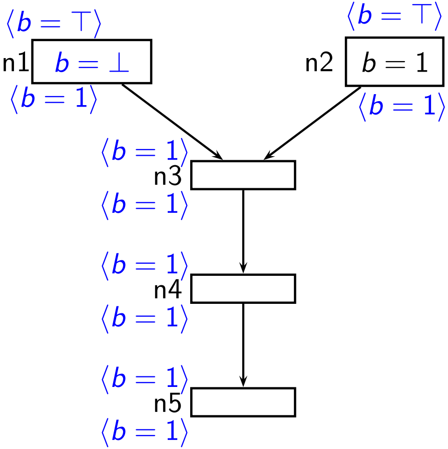
Issues in Constant propagation- Change to Bottom



Issues in Constant propagation- Change to Bottom

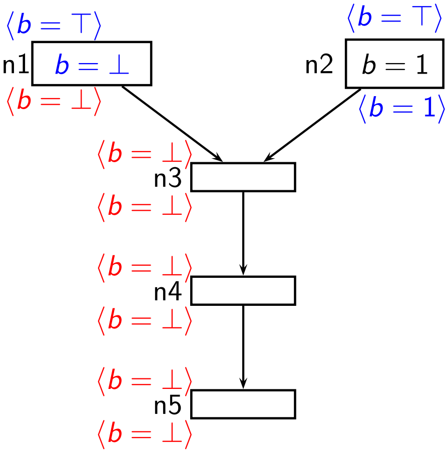


Issues in Constant propagation- Change to Bottom



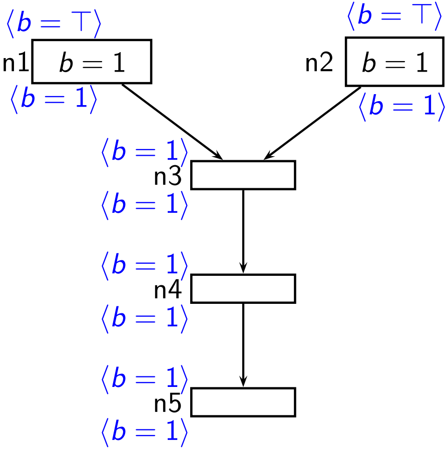
Change to bottom

Issues in Constant propagation- Change to Bottom

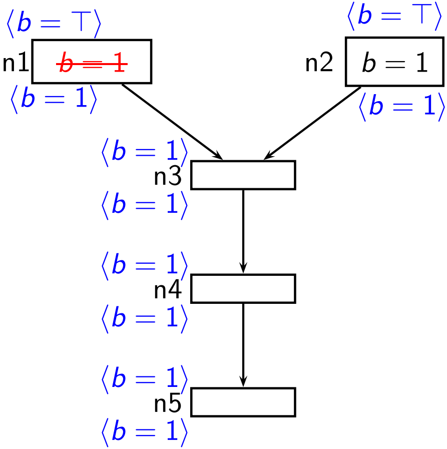

$\langle b = \top \rangle$
n1
$b = \bot$
$\langle b = \bot \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
$\langle b = \bot \rangle$

Directly propagate the change to its neighbour

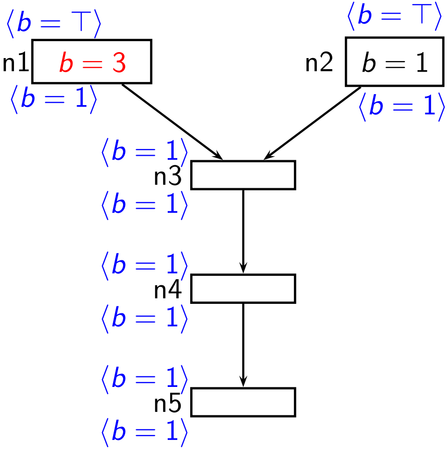
Issues in Constant propagation- Change to other value



Issues in Constant propagation- Change to other value

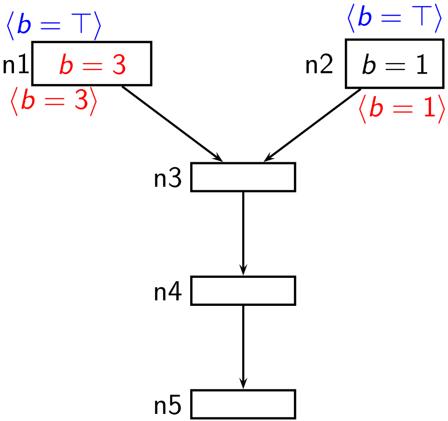


Issues in Constant propagation- Change to other value



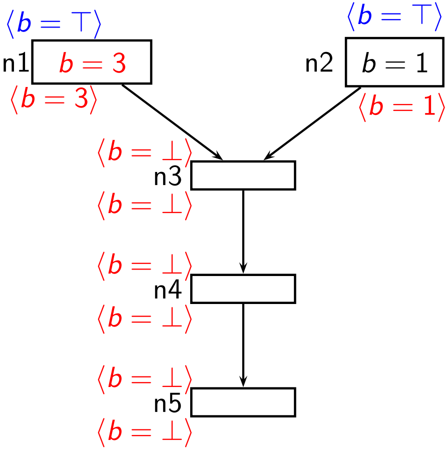
Side Level change

Issues in Constant propagation- Change to other value



Depends on value at OUT_2

Issues in Constant propagation- Change to other value



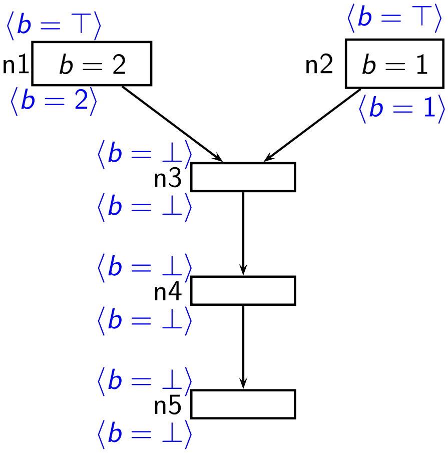
$\langle b = \top \rangle$
n1
$b = 3$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \perp \rangle$
n3
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n4
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n5
$\langle b = \perp \rangle$

Depends on value at OUT_2

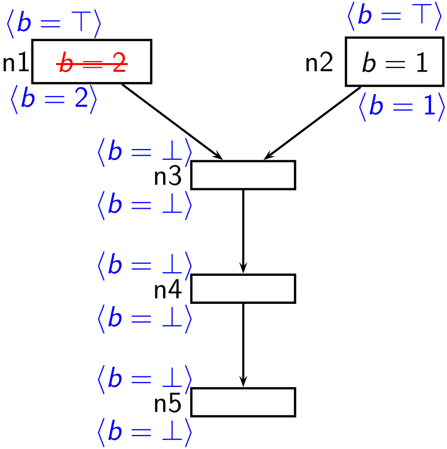
Issues in Constant propagation

- Bottom functions corresponds to reading values from input
    - Changing a flow functions in a node s.t it becomes a bottom function is expected to be rare
- Bottom is also produced due to meet operation
    - May be possible to restrict the size of affected region

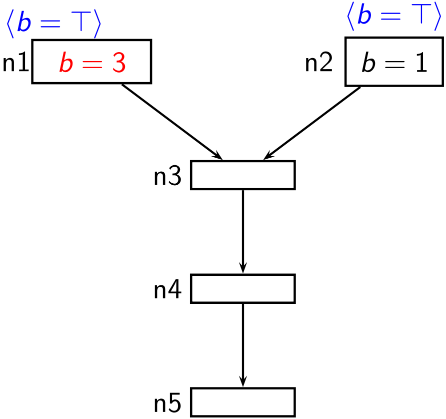
Restrict the size of affected region


$\langle b = \top \rangle$
n1
$b = 2$
$\langle b = 2 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
$\langle b = \bot \rangle$

Restrict the size of affected region

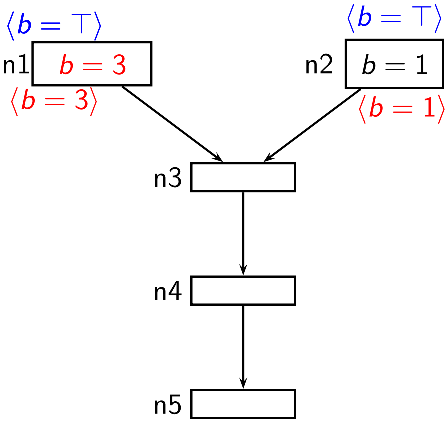

$\langle b = \top \rangle$
n1
~~$b = 2$~~
$\langle b = 2 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
$\langle b = \bot \rangle$

Restrict the size of affected region

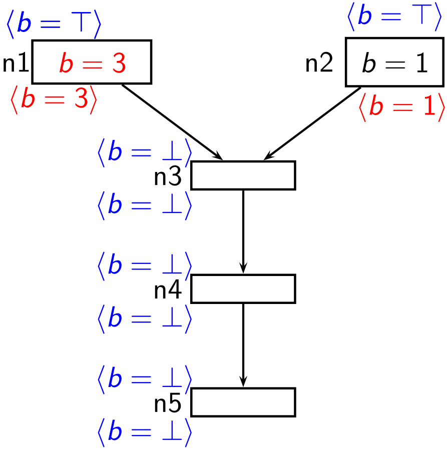


Side Level Change

Restrict the size of affected region

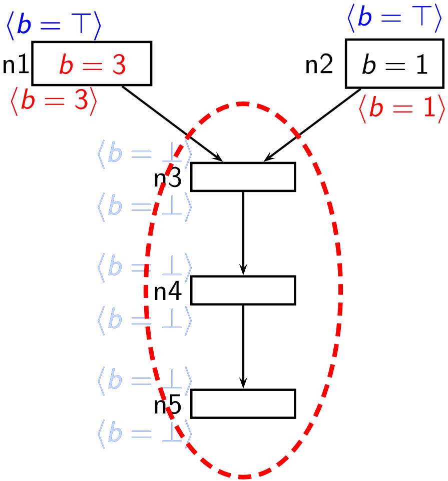


Restrict the size of affected region


$\langle b = \top \rangle$
n1
$b = 3$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \perp \rangle$
n3
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n4
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n5
$\langle b = \perp \rangle$

No need to create affected region

Restrict the size of affected region


$\langle b = \top \rangle$
n1
$b = 3$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
$\langle b = \bot \rangle$

No need to create affected region

Part IV

Overview of PRISM

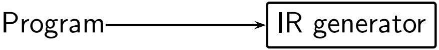
PRISM

- PRISM is a program analyzer generator developed by TATA Research Development and Design Center (TRDDC)

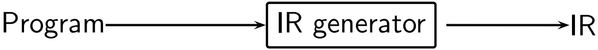
Architecture of PRISM

Program

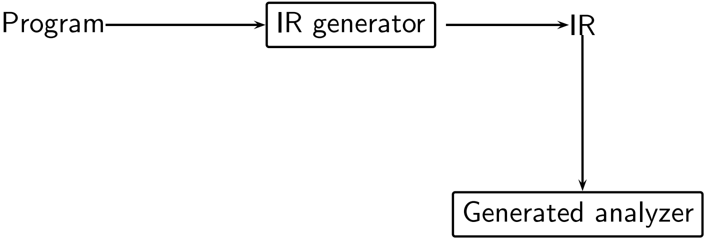
Architecture of PRISM



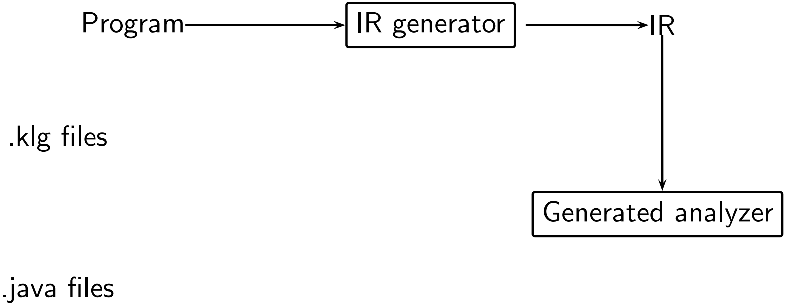
Architecture of PRISM



Architecture of PRISM



Architecture of PRISM



Program

IR generator

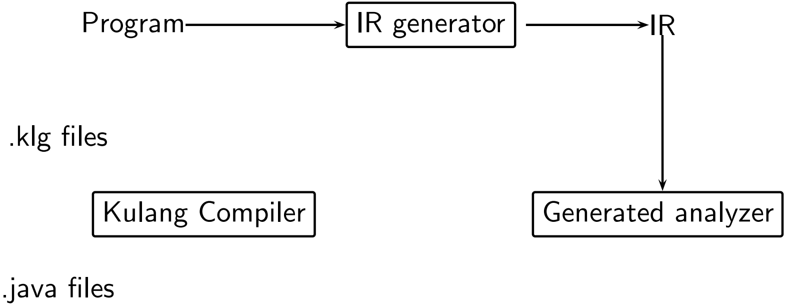
IR

.klg files

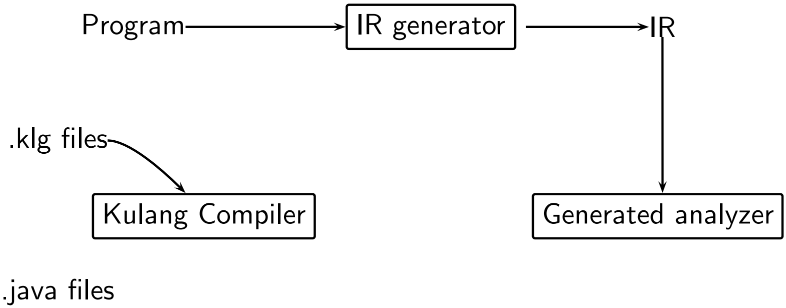
Generated analyzer

.java files

Architecture of PRISM



Architecture of PRISM



Program

IR generator

IR

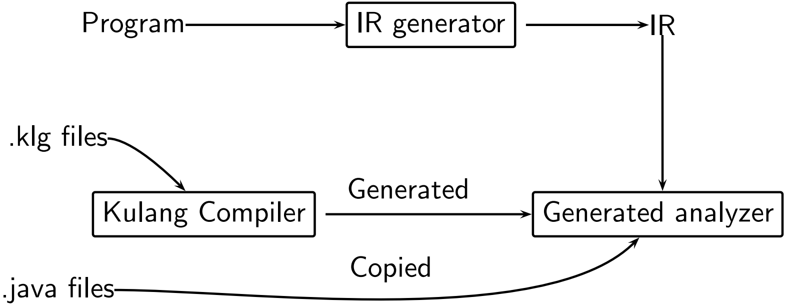
.klg files

Kulang Compiler

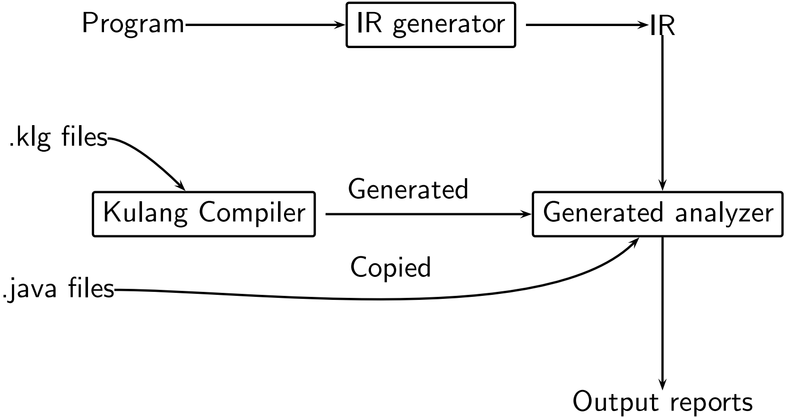
Generated analyzer

.java files

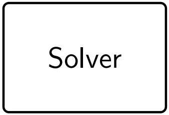
Architecture of PRISM



Architecture of PRISM

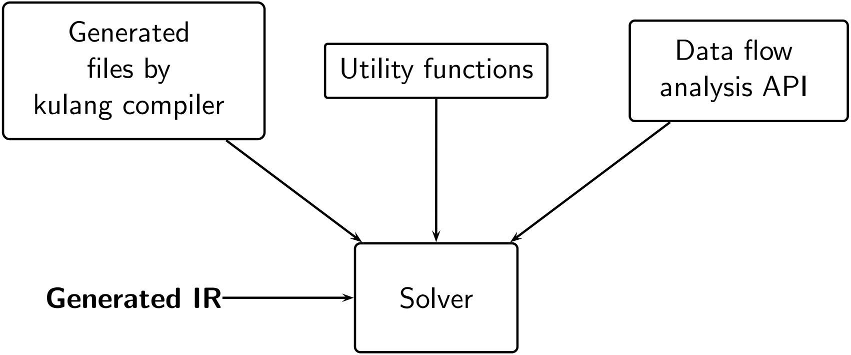


Architecture of analyser generator

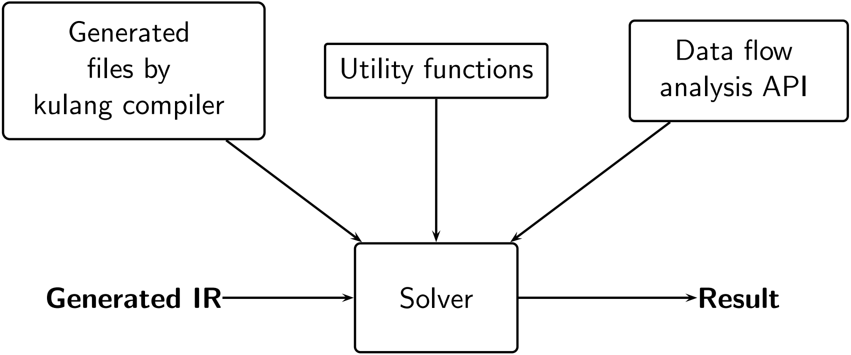


Solver

Architecture of analyser generator



Architecture of analyser generator



Kulang specifications

- Current Kulang specifications accepts:
  - Forward and backward flow functions and meet functions
  - Forward and backward lattice types
  - Forward and backward boundary values and top values

Kulang specifications

```
Forwardlattice Rec ::  res; // lattice types
Backwardlattice Liv :: livenesslattice;

//types of forward and backward lattices
ForwardTop : (res){};
BackwardTop : (livenesslattice){};

A ForwardMeet B : Meet(A,B); //meet functions
A BackwardMeet B : A+B;

ForwardBoundaryValue : (res){}; // boundary values
BackwardBoundaryValue :(livenesslattice){} ;

// Specification of backward flow functions
BackwardNodeflow( n: Binary, R: Rec, L:Liv )
```

Part V

Liveness-based Reaching Definition analysis

Reaching Definition analysis

- In order to understand PRISM and Kulang specification- implemented Reaching Definition analysis for both with and without Liveness

Data flow equations for Liveness-based Reaching Definition analysis

$$LIn_n = f_n(Out_n)$$

$$LOut_n = \begin{cases} BI & \text{n is End} \\ \bigcup\limits_{s \in succ(n)} In_s & \text{otherwise} \end{cases}$$

where,

$$f_n(X) = \begin{cases} (X - \{y\}) \cup (Opd(e) \cap Var) & \text{n is y} = \text{e, e} \in \text{Expr, y} \in \text{X} \\ X - y & \text{n is input(y)} \\ X \cup y & \text{n is use(y)} \\ X & \text{otherwise} \end{cases}$$

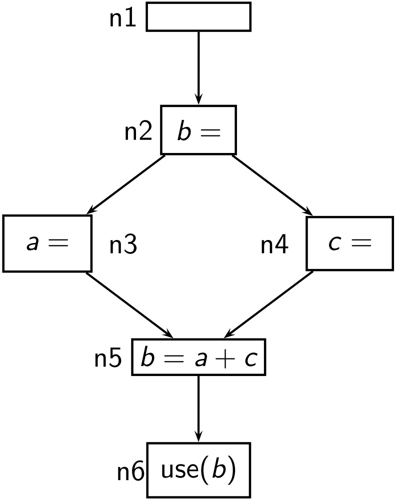
Data flow equations for Liveness-based Reaching Definition analysis

$$RIn_n = \begin{cases} RBI & \text{n is Start block} \\ \bigcup\limits_{p \in pred(n)} Out_p \mid_{LIn_n} & \text{otherwise} \end{cases}$$

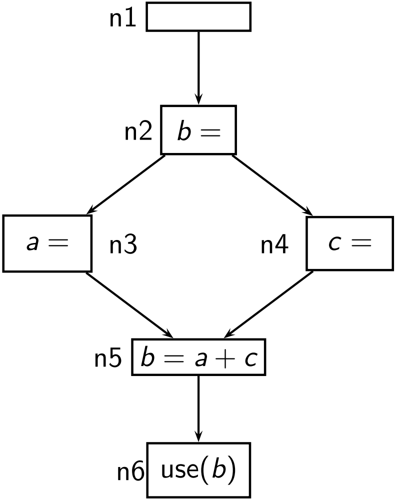
$$ROut_n = Gen_n \cup (In_n - Kill_n) \mid_{LOut_n}$$

$$RBI = \{d_x : x = undef \mid x \in Var\}$$

Example for Liveness-based Reaching Definition

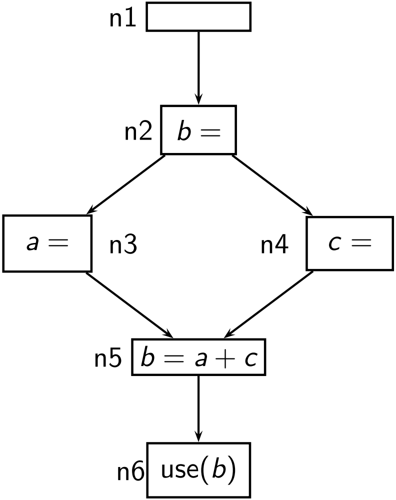


Example for Liveness-based Reaching Definition



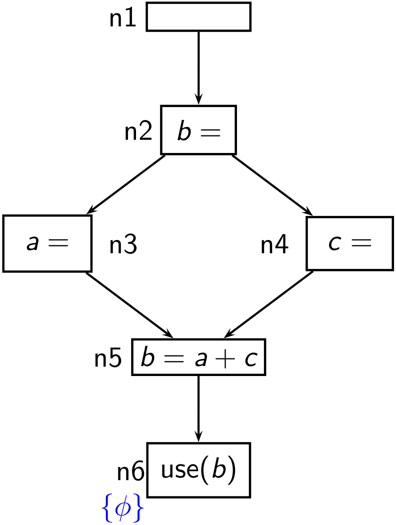
Example for Liveness-based Reaching Definition

Strongly Liveness Pass



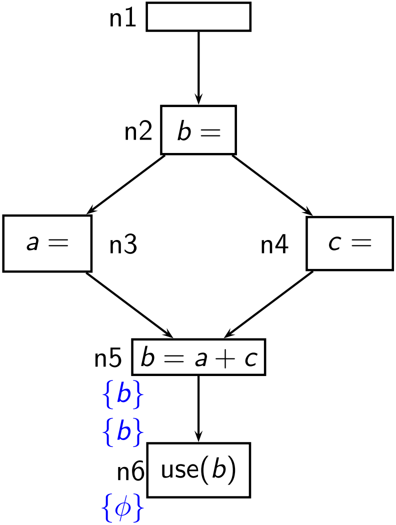
Example for Liveness-based Reaching Definition

Strongly Liveness Pass



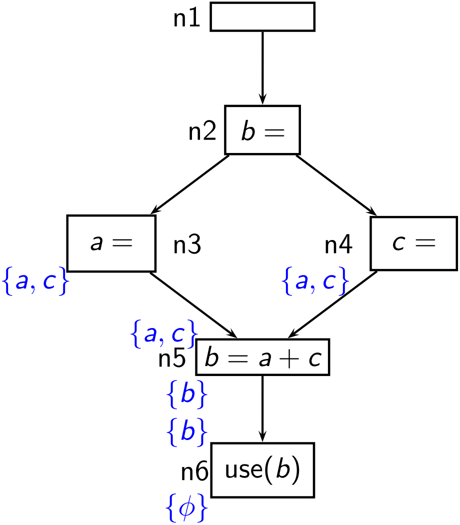
Example for Liveness-based Reaching Definition

Strongly Liveness Pass



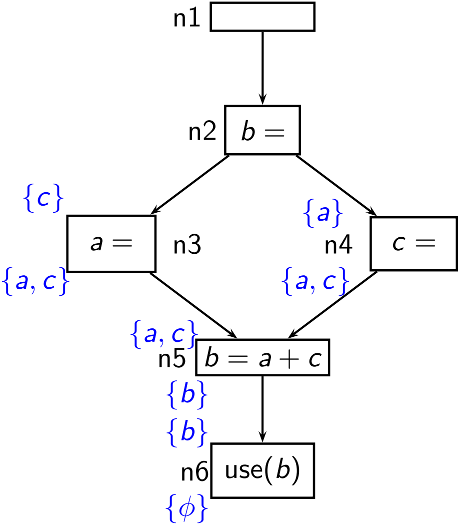
Example for Liveness-based Reaching Definition

Strongly Liveness Pass

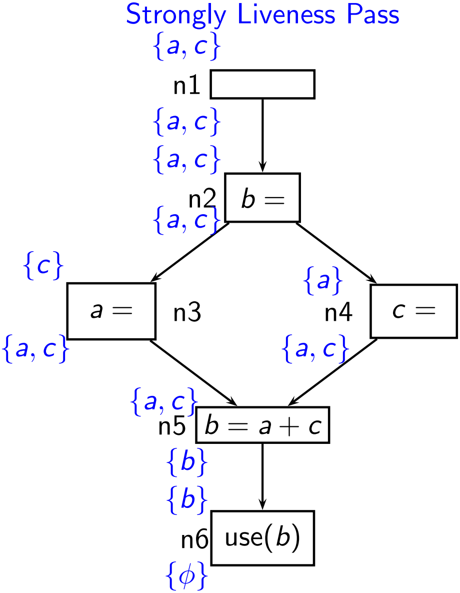


Example for Liveness-based Reaching Definition

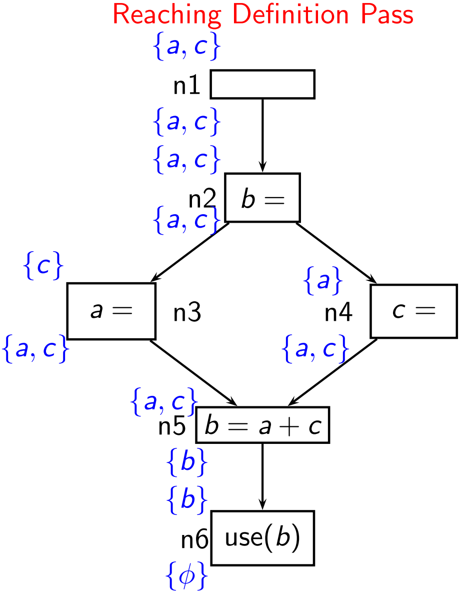
Strongly Liveness Pass



Example for Liveness-based Reaching Definition

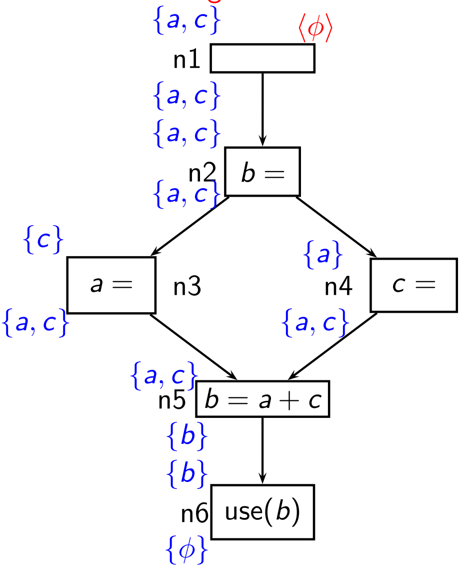


Example for Liveness-based Reaching Definition

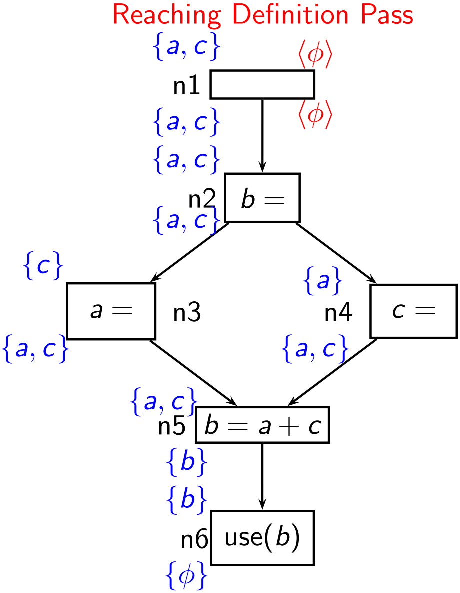


Example for Liveness-based Reaching Definition

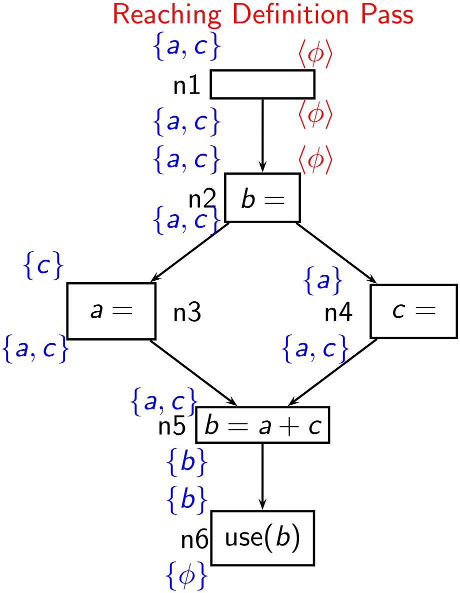
Reaching Definition Pass



Example for Liveness-based Reaching Definition

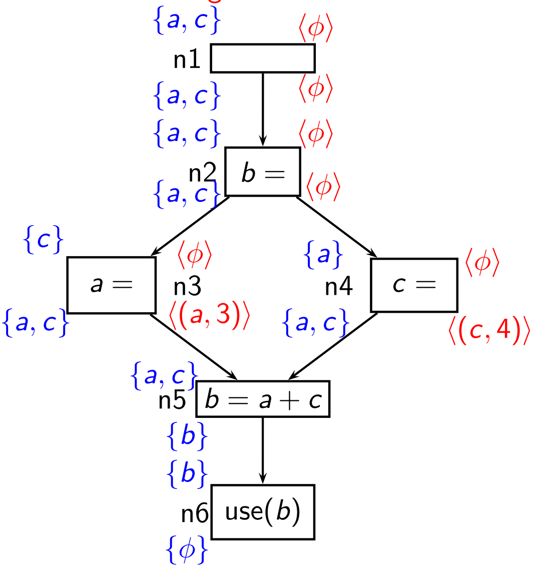


Example for Liveness-based Reaching Definition

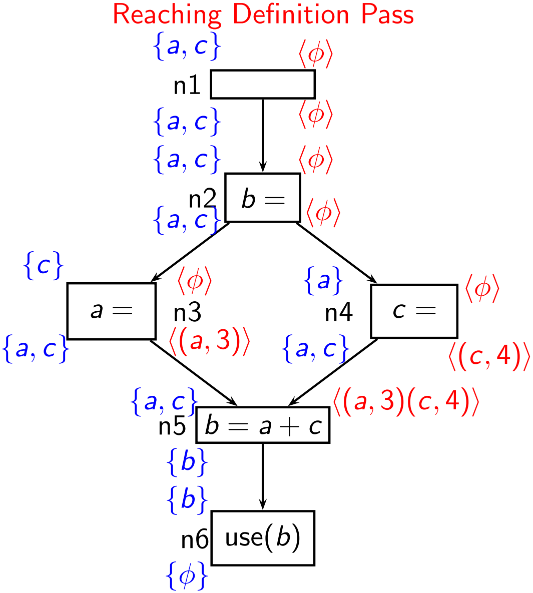


Example for Liveness-based Reaching Definition

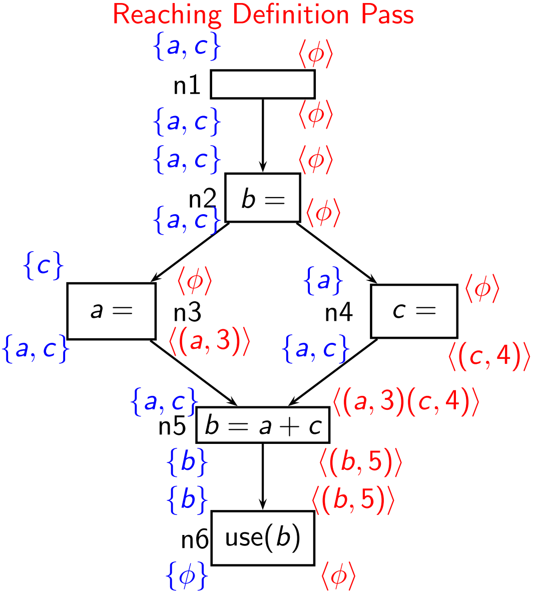
Reaching Definition Pass



Example for Liveness-based Reaching Definition



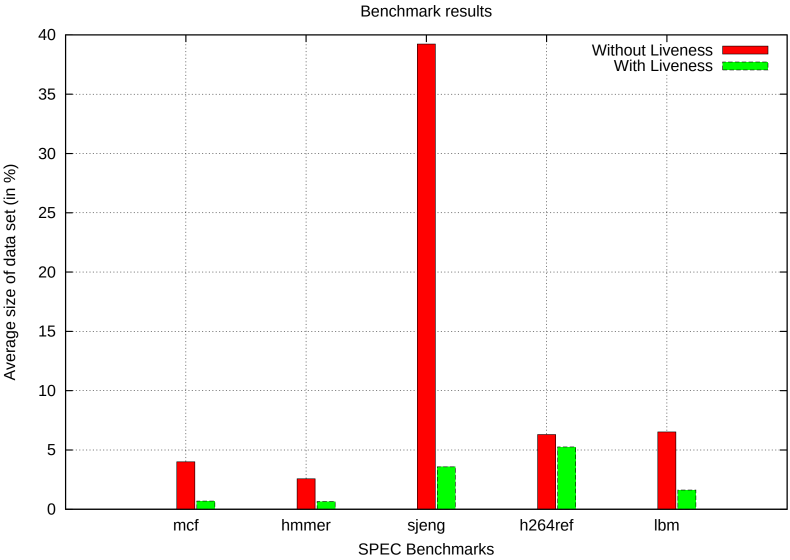
Example for Liveness-based Reaching Definition



Part VI

Performance Measurement

Percentage reduction in size of data set



Limitations of current version of PRISM

- PRISM does not perform incremental analysis

Limitations of current version of PRISM

- PRISM does not perform incremental analysis
- Meet function needs to be explicitly defined in kulang specifications
- The meet function can be inferred from the lattice of the data flow problem

Limitations of current version of PRISM

- PRISM does not perform incremental analysis
- Meet function needs to be explicitly defined in kulang specifications
- The meet function can be inferred from the lattice of the data flow problem
- There is no proper way to debug the kulang specifications

Limitations of current version of PRISM

- PRISM does not perform incremental analysis
- Meet function needs to be explicitly defined in kulang specifications
- The meet function can be inferred from the lattice of the data flow problem
- There is no proper way to debug the kulang specifications
- Hard to understand the specification language

Part VII

Future Work

Future Work

- Method to restrict the size of affected region in Constant Propagation

Future Work

- Method to restrict the size of affected region in Constant Propagation
- Incremental analysis in PRISM

Future Work

- Method to restrict the size of affected region in Constant Propagation
- Incremental analysis in PRISM
- Simplified Kulang specification

Part VIII

Thank You !